LAB 1

MỤC TIÊU

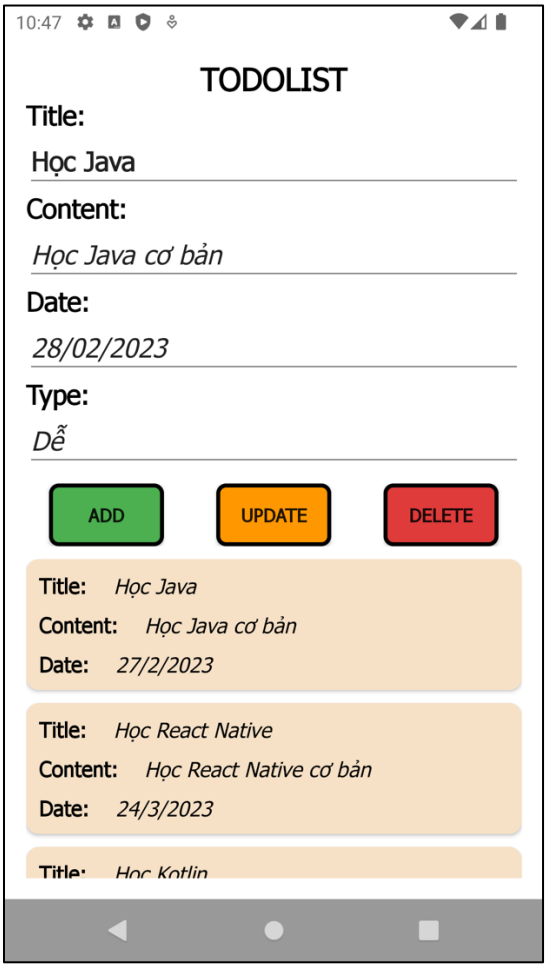
Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Sử dụng SQLite

NỘI DUNG

Tạo 1 ứng dụng ToDo List với các chức năng xem danh sách những việc cần làm, thêm 1 việc cần làm vào danh sách vào database SQLite

Tạo một màn hình có giao diện như sau:



BÀI 1: Khởi tạo database SQLite

Bước 1: Tạo một file mới đặt tên nó là **DbHelper**. Trong file này ta sẽ extends **SQLiteOpenHelper**, sau đó tạo constructor cho class và implements các phương thức onCreate, onUpgrade

```
public class DbHelper extends SQLiteOpenHelper {
    1 usage
    public DbHelper(Context context) {
        //Khởi tạo database có tên là TodoDatabase và version là 1
        super(context, name: "TodoDatabase", factory: null, version: 1);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {...}

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {...}
}
```

Bước 2: Tại hàm **onCreate** khởi tạo table và thêm một số data cho table vừa tạo

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    // Tạo câu lệnh tạo bảng
    String sql = "CREATE TABLE TODO(ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                                    "TITLE TEXT, CONTENT TEXT, DATE TEXT, TYPE TEXT, STATUS INTEGER)";
    sqLiteDatabase.execSQL(sql);

    // Tạo câu lệnh thêm dữ liệu vào database
    String data = "INSERT INTO TODO VALUES(1, 'Học Java', 'Học Java cơ bản', '27/2/2023', 'Bình thường', 1)," +
                                "(2, 'Học React Native', 'Học React Native cơ bản', '24/3/2023', 'Khó', 0)," +
                                "(3, 'Học Kotlin', 'Học kotlin cơ bản', '1/4/2023', 'Dễ', 0)";
    sqLiteDatabase.execSQL(data);

}
```

Tại hàm **onUpgrade**

```
@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    // Khi upgrade database ta cần kiểm tra version hiện tại version khi ta upgrade có khác nhau hay không
    // Nếu có thực hiện câu lệnh xóa bảng và khởi tạo lại
    if(i != i1)
    {
        // Xóa bảng nếu tồn tại
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS TODO");
        // Gọi lại hàm onCreate
        onCreate(sqLiteDatabase);
    }
}
```

BÀI 2: Thực hiện chức năng hiện thị danh sách công việc lên Listview

Bước 1: Tạo một file mới đặt tên **ToDoDAO**, tạo constructor cho class này

```
public class ToDoDAO {
    1 usage
    private final DbHelper dbHelper;

    1 usage
    public ToDoDAO(Context context) {
        dbHelper = new DbHelper(context);
    }
}
```

Bước 2: Trong ToDoDAO ta tạo một hàm đặt tên là **getListToDo** để lấy toàn bộ danh sách công việc có trong database

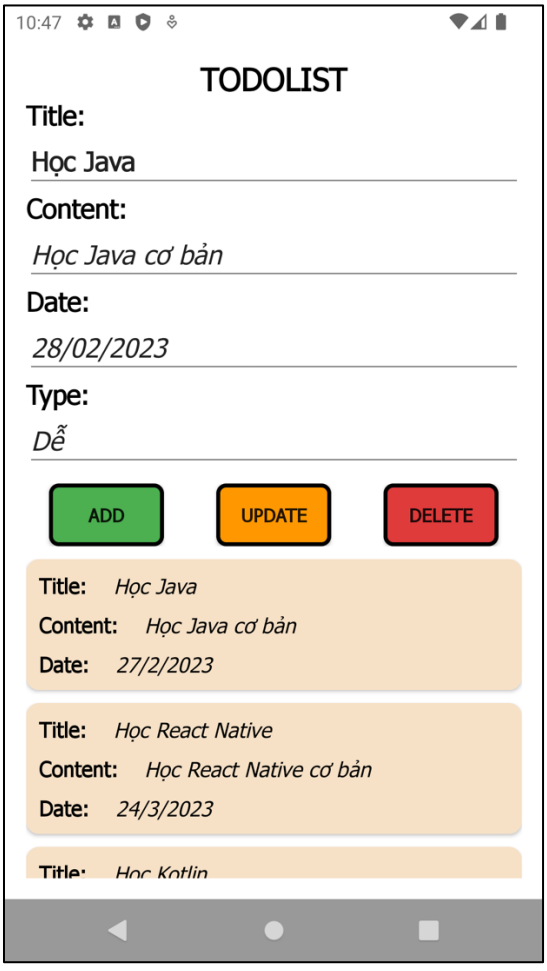
```
public ArrayList<ToDo> getListTodo() {
    //Tạo một danh sách để add dữ liệu Todo vào
    ArrayList<ToDo> list = new ArrayList<>();
    SQLiteDatabase database = dbHelper.getReadableDatabase();
    //Database bắt đầu chạy
    database.beginTransaction();
    try {
        //Tạo câu lệnh truy vấn
        Cursor cursor = database.rawQuery( sql: "SELECT * FROM TODO",  selectionArgs: null);
        if (cursor.getCount() > 0) {
            //Nếu cursor lớn hơn không, di chuyển con trỏ lên đầu
            cursor.moveToFirst();
            //Khởi tạo vòng lặp do..while để lấy toàn bộ dữ liệu truy vấn được thêm vào danh sách
            do {
                list.add(new ToDo(cursor.getInt( columnIndex: 0),
                        cursor.getString( columnIndex: 1),
                        cursor.getString( columnIndex: 2),
                        cursor.getString( columnIndex: 3),
                        cursor.getString( columnIndex: 4),
                        cursor.getInt( columnIndex: 5)));
            } while (cursor.moveToNext());
            //Database đã chạy thành công
            database.setTransactionSuccessful();
        }
    } catch (Exception e) {
        Log.e(TAG,  msg: "getListTodo: " + e);
    } finally {
        //Kết thúc lệnh chạy
        database.endTransaction();
    }
    return list;
}
```

Bước 3: Tạo adapter hiển thị toàn bộ danh sách công việc lấy ở bước 2 lên Listview

(Sinh viên tự thực hiện)

```
ToDoDAO toDoDao = new ToDoDAO( context: this);
ArrayList<ToDo> listToDo = toDoDao.getListTodo();
listToDoAdapter = new ListToDoAdapter( context: this, listToDo);
listTodo.setAdapter(listToDoAdapter);
```

Kết quả ta được:



BÀI 3: THỰC HIỆN CHỨC NĂNG THÊM MỘT CÔNG VIỆC MỚI

Bước 1: Tại file ToDoAO ta tạo thêm một phương thức mới để thêm một công việc vào database

```
public boolean addToDo(ToDo toDo) {
    SQLiteDatabase database = dbHelper.getWritableDatabase();
    database.beginTransaction();
    //Ta sẽ sử dụng ContentValues để đưa dữ liệu vào database
    ContentValues values = new ContentValues();
    //Ở đây id là giá trị tự động tăng nên chúng ta không cần thêm vào
    values.put("TITLE", toDo.getTitle());
    values.put("CONTENT", toDo.getContent());
    values.put("DATE", toDo.getDate());
    values.put("TYPE", toDo.getType());
    values.put("STATUS", toDo.getStatus());
    //Nếu thành công sẽ trả về giá trị tượng ứng số hàng mà dữ liệu được add trong bảng
    long check = database.insert( table: "TODO",  nullColumnHack: null, values);
    return check != -1;
}
```

Bước 2: Tạo sự kiện click vào nút Thêm, gọi đến phương thức addToDo ở bước 1 để thêm 1 công việc vào database.

Chức năng chỉnh sửa, xóa công việc, thực hiện tương tự.

BÀI 4: GIẢNG VIÊN CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết ---